RECEIVED
CENTRAL FAX CENTER

JUN 05 2006

PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission: 28

Application Number	09/783,250
Filing Date	02/14/2001
First Named Inventor	Kallol Pal
Art Unit	2192
Examiner Name	Chuck O. Kendall
Attorney Docket Number	JP920000411US1

ENCLOSURES *(Check all that apply)*

- [x] Fee Transmittal Form
- [x] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
  - [ ] Landscape Table on CD
- [ ] After Allowance Communication to TC
- [ ] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [ ] Other Enclosure(s) (please identify below):

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Law Office of Anthony England		
Signature	Anthony V S England		
Printed name	Anthony V.S. England		
Date	6-5-2006	Reg. No.	35,129

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature	Anthony V S England		
Typed or printed name	Anthony V.S. England	Date	6-5-2006

RECEIVED
CENTRAL FAX CENTER
JUN 05 2006

PTO/SB/17I (11-04)
Approved for use through 07/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PROCESSING FEE Under 37 CFR 1.17(i) TRANSMITTAL

(Fees are subject to annual revision)

Send completed form to: Commissioner for Patents
P.O. Box 1450, Alexandria, VA 22313-1450

Application Number	09/783,250
Filing Date	02/14/2001
First Named Inventor	Kallol Pal
Art Unit	2192
Examiner Name	Chuck O. Kendall
Attorney Docket Number	JP920000411US1

Enclosed is a paper filed under 37 CFR 41.20(b)(2) **that requires a processing fee (37 CFR 1.17(i)).**
Payment of $ 500.00 **is enclosed.**

This form should be included with the above-mentioned paper and faxed or mailed to the Office using the appropriate Mail Stop, if applicable. *For transmittal of petition fees under 37 CFR 1.17(f), (g) or (h), see form PTO/SB/17p.*

Payment of Fees (small entity amounts are NOT available for the petition fees)

☒ The Commissioner is hereby authorized to charge the following fees to Deposit Account No. 09-0457:
☒ processing fee under 37 CFR 1.17(i) ☒ any deficiency of fees and credit of any overpayments
Enclose a duplicative copy of this form for fee processing.

☐ Check in the amount of $_____ is enclosed.

☐ Payment by credit card (Form PTO-2038 or equivalent enclosed). Do not provide credit card information on this form.

Processing Fees under 37 CFR 1.17(i): Fee $130 Fee Code 1808 for all,
Except for § 1.221 papers (Fee Code 1803)

For papers filed under:
§ 1.28(c)(3) - for processing a non-itemized fee deficiency based on an error in small entity status.
§ 1.41 - for supplying the name or names of the inventor or inventors after the filing date without an oath or declaration as prescribed by § 1.63, except in provisional applications.
§ 1.48 - for correcting inventorship, except in provisional applications.
§ 1.52(d) - for processing a nonprovisional application filed with a specification in a language other than English.
§ 1.53(b)(3) - to convert a provisional application filed under § 1.53(c) into a nonprovisional application under § 1.53(b).
§ 1.55 - for entry of late priority papers.
§ 1.99(e) - for processing a belated submission under § 1.99.
§ 1.103(b) - for requesting limited suspension of action, continued prosecution application (§ 1.53(d)).
§ 1.103(c) - for requesting limited suspension of action, request for continued examination (§ 1.114).
§ 1.103(d) - for requesting deferred examination of an application.
§ 1.217 - for processing a redacted copy of a paper submitted in the file of an application in which a redacted copy was submitted for the patent application publication.
§ 1.221 - for requesting voluntary publication or republication of an application. **Fee Code 1803**
§ 1.291(c)(5) - for processing a second or subsequent protest by the same real party in interest.
§ 1.497(d) - for filing an oath or declaration pursuant to 35 U.S.C. 371(c)(4) naming an inventive entity different from the inventive entity set forth in the international stage.
§ 3.81 - for a patent to issue to assignee, assignment submitted after payment of the issue fee.

Anthony V.S. England
Signature

6-5-2006
Date

Anthony V.S. England
Typed or printed name

35,129
Registration No., if applicable

RECEIVED
CENTRAL FAX CENTER
JUN 05 2006

Docket JP920000411US1

Appl. No.: 09/783,250
Filed: February 14, 2001

In the United States Patent and Trademark Office

In re the application of: Kallol Pal	)	
	)	
Filed 02/14/2001	)	Group Art Unit: 2122
	)	
For: Software Testing	)	Examiner: Chuck O. Kendall
	)	
Appl. No.: 09/783,250	)	
	)	
Appellant's Docket: JP920000411US1	)	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

REAL PARTY IN INTEREST

The assignee, International Business Machines Corporation, is the real party in interest.

RELATED APPEALS AND INTERFERENCES

This is the second appeal in the present patent application. There are no other appeals or interferences known to the appellant or its legal representative. International Business Machines Corporation is the sole assignee of the patent application.

06/06/2006 TL0111 00000059 090457 09783250
01 FC:1402 500.00 DA

STATUS OF CLAIMS

Claims 3, 5-17, 20, 22-30, 33, and 35-45 are pending in the application. (Claims 1-45 were originally submitted. Claims 1, 2, 4, 8, 18, 19, 21, 31, 32, and 34 were subsequently canceled.) All the pending claims stand rejected. Office action of December 30, 2005 (the "Final Office Action"). Appellant has appealed from the final rejection. Notice of Appeal, sent by facsimile transmission to the USPTO on April 4, 2006.

The claims appealed herein, and for which arguments are herein presented, are claims 13, 29 and 42. (Arguments are *not* herein presented regarding claims 3, 5-12, 14-17, 20, 22-28, 30, 33, 35-41, and 43-45. However, Appellant contends, of course, that these claims are allowable since they depend on claims for which arguments are herein presented and which Appellant contends are allowable.)

History of the Case

In a first Office action dated September 25, 2003, all the claims were rejected. Specifically, claims 1-6, 9-15, 17-23, 25-36 and 38-44 were rejected under 35 U.S.C. 102(e) based on U.S. patent 6,067, 639 (Rodrigues). Claims 7, 8, 24, 37 and 45 were rejected under 35 U.S.C. 103(a) based on Rodrigues in view of U.S. patent 5,860,009 (Uchihira). Claim 16 was rejected under 35 U.S.C. 103 (a) based on Rodrigues et al. in view of U.S. patent 6,397,378 B1 (Grey). In Amendment A, dated December 26, 2003, method claims 1, 2, 3 and 5, computer program product claims 18, 19, 20 and 22, and system claims 31, 32, 33 and 35 were responsively amended in an effort to overcome the rejections. Claims 23 and 36 were also amended to conform them to their amended base claims. Claims 9, 25 and 38 were amended to correct informalities.

In a second Office action of March 10, 2004, claims 1-45 were finally rejected. Specifically, claims 1-6, 9-16, 17-23, 25-36 and 38-44 were rejected under 35 U.S.C. 103(a) based on Rodrigues in view of U.S. patent 6,067,639 (Darty). Claims 7, 8, 24, 37 and 45 were rejected under 35 U.S.C. 103(a) based on Rodrigues in view of Darty, and further in view of Uchihira. Appellant filed a Notice of Appeal of June 9, 2004, then an Appeal Brief of August 14, 2004.

In a third Office action of December 7, 2004, the Examiner reopened prosecution, withdrawing the Rodrigues reference and asserting a new reference, Telcordia Software Visualization and Analysis Toolsuite User's Manual, Chapter 3, ATAC: Overview (ATAC). The third Office action stated claims 13, 29, & 42 were allowable if written in independent form incorporating all the elements and limitations of their respective base claims. Inexplicably, the third Office action was final.

Appellant filed a Request for Reconsideration on January 4, 2005, within the first month of the reply period, to contend the final rejection was improper and request a new, non-final Office action.

However, upon further consideration Appellant filed a Supplemental Reply on March 28, 2005, which was during the fourth month of the reply period, withdrawing the Request for Reconsideration, canceling the unallowed claims and amending claims to put them in condition for allowance, exactly as indicated in the third Office action. The Supplemental Reply of March 28, 2005, was before any reply from the Office to Appellant's the Request for Reconsideration. In reply, an Advisory Action of April 6, 2005, reversed the Office's previous position and did not allow the amended claims. The rejection of the Advisory Action relied only on the art that had already been cited in the indicated allowability of the third Office action.

In reply to the Advisory Action, Appellant requested in a written Request for Action of June 2, 2005, and in numerous telephone conversations prior to that written request, that a new, non-final Office action be issued, and that the period for reply be restarted in the new Office action.

A fourth Office action of June 8, 2005, which was non-final, essentially repeated the rejections of the third Office action, but also reapplied Grey in combination with certain ones of the other references for rejection of some of the claims.

Appellant filed a Reply to Office action on October 7, 2005, presenting arguments to traverse, but did not amend the claims.

The present, fifth Office action (the "Final Office action") December 30, 2006, maintains the same rejections.

Specific Rejection In the Present Office Action

Claims 3, 5, 6, 9, 13, 15, 16, 20, 22, 23, 25 and 29 stand rejected under 35 USC 103(a) as being unpatentable over Darty, in view of ATAC. Claims 10-12, 14, 17, 26-28 and 30 stand rejected under 35 USC 103(a) as being unpatentable over Darty, in view of ATAC, and further in view of Rodrigues. Claims 7, 8 and 24 stand rejected under 35 USC 103(a) as being unpatentable over Darty, in view of ATAC, and further in view of Uchihira. Claims 33, 35-38, 40-43 and 45 stand rejected under 35 USC 103(a) as being unpatentable over Darty, in view of ATAC, and further in view of Grey. Claims 39 and 44 stand rejected under 35 USC 103(a) as being unpatentable over Darty, in view of ATAC, Grey, and Rodrigues.

STATUS OF AMENDMENTS

There are no amendments in connection with this appeal and none were submitted subsequent to the Final Office Action. The claims in the Claim Appendix herein set out the claims as amended in Appellant's reply of October 7, 2005, which was prior to the Final Office Action.

SUMMARY OF CLAIMED SUBJECT MATTER

The present invention is claimed in the form of a method, a computer program product and a system in independent claims 13, 29, and 42, respectively.

Claim 13

Claim 13 points out that a method of testing a program includes step a), according to which the program is divided into groups such that every statement in the program belongs to at least one of the groups. See present application, page 18, lines 1-2, 8-15, and FIG's 4A - 4C (showing all statements in the program of FIG. 4A, i.e., lines 3, 5, 6, 8 - 18, 20 - 25, 27 and 28, divided into respective groups B11 - B18 in FIG's 4B and 4C, wherein group B11 in FIG. 4B includes lines corresponding to lines 3, 5 and 6 of FIG. 4A, group B12 includes lines corresponding to lines 8 and 9 of FIG. 4A, group B13 in FIG. 4B includes a line corresponding to line 10 of FIG. 4A, group B14 in FIG's 4B and 4C includes lines corresponding to lines 11 -

13 of FIG. 4A, group B15 in FIG. 4C includes lines corresponding to lines 14 - 15 of FIG. 4A, group B16 in FIG. 4C includes lines corresponding to lines 16 - 18 of FIG. 4A, group B17 in FIG. 4C includes lines corresponding to lines 20 - 21 of FIG. 4A, and group B18 in FIG. 4C includes lines corresponding to lines 22 - 25, 27 and 28 of FIG. 4A).

Claim 13 goes on to state that each of the groups contains a respective sequence of ones of the statements such that all the statements of such a group are executed if at least one statement of said group is executed. See present application, page 8, line 19-page 9, line 2. Also see page 18, lines 2-6, and FIG's 4A - 4C (showing sequences such as the statement of line 10 of FIG. 4A for group B13 (a sequence of one statement) and the statements of lines 11 - 13 of FIG. 4A for group B14, etc.).

Claim 13 goes on to state in that such a group is deemed to be executed if at least one of the statements of the group is executed when the program is executed. See present application, page 18 and FIG's 4A - 4C (showing groups of statements in which none of the groups have any branching statements, except at the end of the group, that would cause some of the statements in the group to not be executed along with the others in the group).

Claim 13 goes on to set out step b), according to which there is a determining of the ones of the groups that are executed when said program is executed while testing said program. See present application, page 13, lines 10-12, and FIG. 2.

In step c) Claim 13 states unexecuted ones of the groups are indicated based on the ones of the groups that were determined in step b) to have been executed. See present application, page 14, lines 1-7, and FIG. 2.

In step d), claim 13 states a tester is enabled to execute said unexecuted groups such that said tester can ensure that all statements in said program are executed at least once. See present application, page 14, lines 8-11, and FIG. 2.

Step e) of Claim 13 states that step d) includes an extra statement in each of said groups, wherein execution of such an extra statement enables said determining in step b) to identify an executed one of the groups corresponding to said extra statement, wherein said program is contained in a plurality of programs which in turn are contained in a class of an object oriented environment. See present application, page 4, lines 2-9. See also page 15, lines 9-12, and FIG. 3.

In step f), claim 13 states that the tester is enabled to define a macro containing a plurality of program lines and storing said macro in a database. See present application, page 9, lines 21 - page 10, line 3. See also page 17, 7-12, and FIG. 3.

In step g), claim 13 states that the tester is enabled to execute said macro in the middle of testing said plurality of programs. See present application, page 28, lines 20-25, and FIG. 11A.

Claim 29

Claim 29 sets out computer readable program code means that includes a number of constituent means. Regarding the computer readable program code means generally, see present application, page 12, lines 5-20 (describing that processes described in the patent application may be implemented as instructions embodied in computer readable program code means for execution by a computer system).

Regarding the dividing means of the computer readable program code means in claim 29, see description herein above of process in step a) of claim 13.

Regarding the determining means in claim 29, see description herein above of process in step b) of claim 13.

Regarding the indicating means in claim 29, see description herein above of process in step c) of claim 13.

Regarding the first enabling means in claim 29, see description herein above of process in step d) of claim 13.

Regarding the means for including in claim 29, see description herein above of process in step e) of claim 13.

Regarding the second enabling means in claim 29, see description herein above of process in step f) of claim 13.

Regarding the third enabling means in claim 29, see description herein above of process in step g) of claim 13.

Regarding the storing means in claim 29, see present application, page 15, lines 4-7.

Claim 42

Claim 42 sets out a system enabling a tester to test a program having statements, said computer system. The claimed system includes a random access memory 120, a display unit 170, an input interface 190, a processor (110), and a storing means (secondary memory 130, which may include hard drive 135 and removable storage unit 137. Present application, FIG. 1, and page 11, lines 3-4, page 12, lines 1-2. Secondary memory 130 is suitable for storing a macro or the like. Present application, page 17, lines 8-9.

Claim 42 goes on to state various actions performed in connection with execution of a program by the processor. Regarding dividing said program into a plurality of groups, etc., as stated in claim 42, see description herein above of process in step a) of claim 13.

Regarding the processor determining ones of the groups that are executed, etc., as stated in claim 42, see description herein above of process in step b) of claim 13.

Regarding the display indicating, etc., as stated in claim 42, see description herein above of process in step c) of claim 13.

Regarding the processor enabling the tester to execute, etc., as stated in claim 42, see description herein above of process in step d) of claim 13.

Regarding the processor including an extra statement, the execution of the extra statement enabling the processor to identify an executed one of the groups, and the program being contained in a plurality of program which in turn are contained, etc., as stated in claim 42, see description herein above of process in step e) of claim 13.

Regarding the processor receiving program lines representing a macro, as stated in claim 42, see description herein above of process in step f) of claim 13.

Regarding the processor executing the macro, as stated in claim 42, see description herein above of process in step g) of claim 13.

GROUND OF REJECTION TO BE REVIEWED ON APPEAL

Claims 3, 5, 6, 9, 13, 15, 16, 20, 22, 23, 25 and 29 stand rejected under 35 USC 103(a) as being unpatentable over Darty, in view of ATAC. Appellant respectfully submits that the rejection is improper.

ARGUMENTS

Claims 13, 29 and 42

Issue One: Darty does not teach or suggest that for which it is relied upon in the rejection of claims 13, 29 and 42.

All the limitations of the subject claims must be taught or suggested by the art relied upon. MPEP 2143.03 (citing In re Royka, 490 F.2d 981, 180 USPQ 580 (CCPA 1974); In re Wilson, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)). Darty is relied upon for teaching that groups, blocks, or any other such thing, have the property that "every statement in the program belongs to at least one of the groups" so that "indicating unexecuted ones of the groups based on the ones of the groups that . . . have been executed . . . enabl[es] a tester to execute said unexecuted groups such that said tester can ensure that all statements in said program are executed at least once," as claimed in the present case. Appellant respectfully submits that Darty does not teach or suggest this.

In discussing fault isolation, Darty teaches fault isolation is done by correlating "test points" with "dependency sets," which includes a process of determining which dependency sets contain test point failures in progressive succession from larger to smaller dependency sets, and then comparing which sets have failures in order to converge on a "smallest fault isolation set." Darty, column 14, lines 51-67. Darty teaches that a single block of code having a fault might not be conclusively identified by test points. Darty, column 15, lines 54-61; column 18, lines 60-67. Darty states that test points are "strategically placed" so that a block of code that is the "most probable" source of a failure may be identified, "depending on the number and placement of test points." Darty, col. 9, lines 24-38; see also col. 10, lines 3-10.

It appears that Darty concerns detecting whether errors arose in a program that has run or is running, and if an error arises, detecting what triggered the error. For example, Darty states, "At step S153, the method of the present invention inspects the fault isolation set matrix to determine whether any entries exist therein. If the set matrix is empty, the method ends." Darty, column 12, lines 22-25. The issue of whether all statements in the program are executed at least once simply does not arise in the context of Darty's teachings.

i) Darty does not teach or suggest that "every statement in the program belongs to at least one of the groups."

Claim 13 of the present application states that the method includes "dividing said program into a plurality of groups such that *every statement in the program belongs to at least one of the groups*" (emphasis added). (Claims 29 and 42 have similar language, according to the respective forms of the invention they claim.) With regard to element 102 in FIG. 3A, Darty merely teaches that "lines of the program to be tested are grouped into functional blocks." Darty, col. 9, lines 39-41. This does not teach or suggest that *all* lines of the program are grouped into blocks. Indeed, in one arrangement Darty shows a "Program ABC" in Text Layout 2, that is divided into three blocks, wherein some of the statements of the program are clearly and explicitly shown as *not* belonging to any of the three blocks. Darty, column 5, lines 30-40.

The Final Office Action responds with the contention that Darty, col. 21, lines 48-50, teaches *all* lines of the program are grouped into blocks. Final Office Action, page 12, second paragraph. However, this passage in a claim of Darty merely describes "grouping lines of code of the computer program into functional blocks." The Final Office Action also argues that Darty FIG. 5 teaches *all* lines of the program are grouped into blocks. Final Office Action, page 12, second paragraph. However, this figure of Darty merely shows there are blocks.

Darty's statement that "lines of the program to be tested are grouped into functional blocks" may be interpreted as meaning that in the program to be tested, *some, but not all,* lines are grouped into functional blocks. Darty's claim of "grouping lines of code of the computer program into functional blocks" may also be interpreted as meaning that in the program to be tested, *some, but not all,* lines are grouped into functional blocks. Darty's FIG. 5 may be

interpreted as showing the functional blocks into which *some, but not all,* lines are grouped in the program to be tested. Indeed, Darty teaches that all blocks of code do not necessarily have associated test points. Darty, col. 10, lines 7-10. Why, then, is it presumed Darty teaches that all lines are necessarily grouped into blocks, as claimed in the present application, given that Darty never specifically states this?

ii) Darty does not teach or suggest "enabling a tester to execute said unexecuted groups such that said tester can ensure that all statements in said program are executed at least once."

Claim 13 of the present application also states that the method includes "an extra statement in each of said groups, wherein execution of such an extra statement enables said determining in step b) [, i.e., determining the ones of the groups that are executed when said program is executed while testing said program,] to identify an executed one of the groups corresponding to said extra statement." (Regarding the above statement and the following discussion, it should be understood that claims 29 and 42 have similar language, according to the respective forms of the invention they claim.) It follows that identifying the *executed* ones of the groups enables identifying the *unexecuted* groups. In other words, as stated in claim 13 of the present case, the method includes "indicating unexecuted ones of the groups based on the ones of the groups that were determined in step b) to have been executed." It further follows that identifying the unexecuted groups enables *executing* any such unexecuted group. In other words, as stated in claim 13 of the present case, the method includes "enabling a tester to execute said unexecuted groups." Furthermore, in the present case "groups" have the property that "every statement in the program belongs to at least one of the groups." It therefore follows, as a specific consequence of this property, that once any *unexecuted* groups are executed, *all lines in the programs will have been executed at least once*. See, first paragraph of Detailed Description of the Preferred Embodiments of the present application ("A testing program provided according to an aspect of the present invention divides each program in a class into multiple groups, with each group containing a sequence of program lines such that one can be certain that all the program lines of the group will be executed if one of the lines is executed, but for the occurrence of some abnormal condition.

As a tester tests the programs in the classes, the testing program keeps track of the groups which have been executed. By ensuring at least one statement in all groups is executed, the tester can ensure that all lines in the programs are executed at least once.") In other words, as stated in claim 13 of the present case, enabling the tester to execute said unexecuted groups produces a result "such that said tester can ensure that all statements in said program are executed at least once."

The Final Office Action contends that Darty, Figure 3d, steps S150, S153, S155, S160 and S148 teach "enabling a tester to execute said unexecuted groups such that said tester can ensure that all statements in said program are executed at least once," as claimed. Appellant has carefully reviewed Darty's description of Figure 3d, steps S150, S153, S155, S160 and S148, with particular scrutiny on Darty, column 11, line 55 - column 12, line 41, and respectfully submits that Darty simply does not teach or suggest what is claimed.

The Final Office Action interprets "fault isolation set empty" in Darty's Figure 3d, step S153, to mean "no errors/unexecuted code." Final Office Action, page 13, response 2. However, Darty does not mention "no unexecuted code" and no basis is offered for the addition of the condition "no unexecuted code" by this interpretation. The Final Office Action also interprets the repeating of steps depicted in Darty Figure 3d if the fault isolation set is not empty at S153 as "enabling all unexecuted code to be executed at least once." Id. However, Darty does not mention enabling all unexecuted code to be executed at least once, and merely states that when the fault isolation set is empty at S153 this "indicat[es] a successful program execution with no code failures." Darty, col. 12, lines 33-34.

It does not follow from Darty's teaching about "successful program execution with no code failures" that this teaches or suggests what is claimed in the present application, i.e., that "all statements in the program [have] executed at least once."

Issue Two: ATAC does not qualify as prior art with respect to the present patent application.

"Prior art disclosures on the Internet or on an on-line database are considered to be publicly available as of the date the item was publicly posted. If the publication does not include a publication date (or retrieval date), it cannot be relied upon as prior art under 35 U.S.C. 102(a) or (b) . . ." MPEP 2128. The Internet publication that has been provided to

Appellant and referred to as "ATAC" does not have a publication date or even a copyright date. (A copy of the first page of ATAC, as provided to Appellant, is enclosed as Appendix DD.) The November 5, 2004, retrieval date of ATAC is more than three years after the filing date of the present application. Therefore, ATAC does not qualify as prior art with respect to the present patent application.

Issue Three: The new ground of rejection introduced by the ATAC reference is improper because it has not been shown to "fully meet" at least one claim or meet a claim except for differences shown to be "completely obvious." MPEP 706.7(e).

In the Office action of March 10, 2004, i.e., claims 13, 29 and 42, among others, were rejected on the basis of the combination of Darty and Rodrigues. Upon reopening prosecution after the filing of Appellant's prior Appeal Brief, claims 13, 29 and 42 were indicated as allowable if amended to include their base claims. This clearly indicates that claims 13, 29 and 42 are allowable over the combination of Rodrigues and Darty. However, although claims 13, 29 and 42 were responsively amended solely to incorporate all limitations of their base claims, claims 13 and 29 were then rejected on the basis of the combination of Darty and ATAC and claim 42 was rejected on the basis of the combination of Darty, ATAC and Grey. Advisory Action of April 6, 2005, and Office action of June 8, 2005.

When an Examiner has become convinced a previously rejected claim is allowable over prior grounds of rejection, the claim should be allowed, except for certain limited situations in which the claim may be subjected to a new ground of rejection. MPEP 706.7(e) ("The examiner may withdraw the rejection of finally rejected claims. If new facts or reasons are presented such as to convince the examiner that the previously rejected claims are in fact allowable . . ., then the final rejection should be withdrawn. "Although it is permissible to withdraw a final rejection for the purpose of entering a new ground of rejection, this practice is to be limited . . ."). The exception applies, that is, a new grounds of rejection is proper instead of allowance, when a certain test is met. MPEP 706.7(e) (". . . this practice is to be limited to situations where a new reference either *fully meets* at least one claim or meets it except for differences which are shown to be *completely obvious*" (emphasis added)).

Responsive to Appellant's prior Appeal Brief the Examiner became convinced previously rejected claims 13, 29 and 42 were allowable over prior grounds of rejection. Given this, the new grounds of rejection are not proper unless the test of MPEP 706.7(e) is met. Appellant submits that the newly cited reference, ATAC, does not fully meet at least one claim, nor do so except for differences shown to be completely obvious. The present Office action does not even *assert* that ATAC meets this test, and presents no arguments to make this showing. Therefore, the entering of this new ground of rejection is contrary to the stated procedure of MPEP 706.7, cited above, and should be withdrawn.

REQUEST FOR ACTION

Based on the above arguments, Appellant requests that the pending claims of the present application be allowed and that the application promptly be passed to issuance.

Respectfully submitted,

By Anthony V S England

Anthony V.S. England
Registration No. 35,129
Attorney of Record for
IBM Corporation
Telephone: 512-477-7165
a@aengland.com

Attachments: Claims Appendix AA, Evidence Appendix BB, Related Proceedings Appendix CC, ATAC Reference Appendix DD

APPENDIX "AA" CLAIMS

1. (canceled)

2. (canceled)

3. (previously presented) The method of claim 13, wherein said extra statements contain respective group identifiers, wherein said determining in step b) further comprises examining such a group identifier to determine a specific one of the groups which has been executed.

4. (canceled)

5. (previously presented) The method of claim 13, further comprising the steps of:

grouping a sequence of the groups into a block; and

determining that said block has been executed only if all of the groups of the block are executed.

6. (original) The method of claim 5, wherein said grouping comprises:

determining a language structure present in said plurality of programs;

grouping a subset of groups present in said language structure into a block such that the statements in said language structure are presented as a block to said tester.

7. (original) The method of claim 6, wherein said blocks are defined hierarchically according to the inclusive relationship of language structures in said plurality of programs.

8. (original) The method of claim 7, wherein said language structure comprises one of program delimiters, control structure and loop structure.

APPENDIX "AA" CLAIMS

9. (previously presented) The method of claim 13, wherein said enabling comprises:

enabling said tester to examine the statements associated with said unexecuted blocks such that said tester can determine arguments which would cause an unexecuted block to be executed;

enabling said tester to enter said determined arguments to cause said unexecuted block to be executed.

10. (previously presented) The method of claim 9, wherein such an argument comprises an instance of another object.

11. (original) The method of claim 10, further comprises:

enabling said tester to instantiate said instance of said another object;

enabling said tester to assign a name to said instance, wherein said tester can enter said name to provide said instance as an argument value.

12. (original) The method of claim 11, further comprising:

receiving a string as an argument; and

determining that said string indicates that said instance is said argument value if said name matches said string.

13. (previously presented) A method of testing a program having statements, said method comprising the steps of:

a) dividing said program into a plurality of groups such that every statement in the program belongs to at least one of the groups, wherein each of said groups contains a respective sequence of ones of the statements such that all the statements of such a group are executed if at least one statement of said group is executed, and wherein such a group is deemed to be executed if at least one of the statements of the group is executed when the program is executed;

APPENDIX "AA" CLAIMS

b) determining the ones of the groups that are executed when said program is executed while testing said program;

c) indicating unexecuted ones of the groups based on the ones of the groups that were determined in step b) to have been executed;

d) enabling a tester to execute said unexecuted groups such that said tester can ensure that all statements in said program are executed at least once;

e) including an extra statement in each of said groups, wherein execution of such an extra statement enables said determining in step b) to identify an executed one of the groups corresponding to said extra statement, wherein said program is contained in a plurality of programs which in turn are contained in a class of an object oriented environment;

f) enabling said tester to define a macro containing a plurality of program lines;

storing said macro in a database; and

g) enabling said tester to execute said macro in the middle of testing said plurality of programs.

14. (original) The method of claim 13, wherein said macro is designed to examine the data structures within an instance of an object or to set the values for the variables in the object.

15. (previously presented) The method of claim 13, wherein said dividing, determining, indicating and enabling are performed in a single computer system.

16. (previously amended) The method of claim 13, wherein said object is generated in Java Programming language.

17. (previously presented) The method of claim 13, further comprising:

enabling said tester to load said class;

enabling said tester to instantiate an instance of said class; and

enabling said tester to execute said program on said instance.

APPENDIX "AA" CLAIMS

18. (canceled)

19. (canceled)

20. (previously presented) The computer program product of claim 29, wherein said extra statements contain respective group identifiers, wherein said determining means examines such a group identifier to determine a specific one of the groups which has been executed.

21. (canceled)

22. (previously presented) The computer program product of claim 29, further comprising grouping means for grouping a sequence of the groups into a block, and second determining means for determining that said block has been executed only if all of the groups of the block are executed.

23. (previously presented) The computer program product of claim 22, wherein said grouping means comprises:

third determining means for determining a language structure present in said plurality of programs;

a second grouping means for grouping a subset of groups present in said language structure into a block such that the statements in said language structure are presented as a block to said tester.

24. (original) The computer program product of claim 23, wherein said blocks are defined hierarchically according to the inclusive relationship of language structures in said plurality of programs.

APPENDIX "AA" CLAIMS

25. (previously presented) The computer program product of claim 29, wherein said enabling means comprises:

second enabling means for enabling said tester to examine the statements associated with said unexecuted blocks such that said tester can determine arguments which would cause an unexecuted block to be executed;

third enabling means for enabling said tester to enter said determined arguments to cause said unexecuted block to be executed.

26. (previously presented) The computer program product of claim 25, wherein such an argument comprises an instance of another object, and the computer program product further comprises:

means for enabling said tester to instantiate said instance of said another object;

means for enabling said tester to assign a name to said instance, wherein said tester can enter said name to provide said instance as an argument value.

27. (original) The computer program product of claim 26, further comprising:

means for receiving a string as an argument; and

means for determining that said string indicates that said instance is said argument if said name matches said string.

28. (previously presented) The computer program product of claim 29, wherein said macro is designed to examine the data structures within an instance of an object or to set the values for the variables in the object.

29. (previously presented) A computer program product for use with a computer system, said computer program product comprising a computer usable medium having computer readable program code means embodied in said computer usable medium, said computer readable program code means enabling testing of a program having statements, said computer readable program code means comprising:

APPENDIX "AA" CLAIMS

dividing means for dividing said program into a plurality of groups such that every statement in the program belongs to at least one of the groups, wherein each of said groups contains a respective sequence of ones of the statements such that all the statements of such a group are executed if at least one statement of said group is executed, and wherein such a group is deemed to be executed if at least one of the statements of the group is executed when the program is executed;

determining means for determining the ones of the groups that are executed when said program is executed while testing said program;

indicating means for indicating unexecuted ones of the groups
based on said determining;

first enabling means for enabling a tester to execute said unexecuted groups such that said tester can ensure that all statements in said program are executed at least once;

means for including an extra statement in each of said groups, wherein execution of such an extra statement enables said determining means to identify an executed one of the groups corresponding to said extra statement, wherein said program is contained in a plurality of programs which in turn are contained in a class of an object oriented environment;

second enabling means for enabling said tester to define a macro containing a plurality of program lines;

storing means for storing said macro; and

third enabling means for enabling said tester to execute said macro in the middle of testing said plurality of programs.

30. (original) The computer program product of claim 26, further comprising:

means for enabling said tester to load said class;

means for enabling said tester to instantiate an instance of said class; and

means for enabling said tester to execute said program on said instance.

31. (canceled)

APPENDIX "AA" CLAIMS

32. (canceled)

33. (previously presented) The system of claim 42, wherein said extra statements contain respective group identifiers, wherein said processor examines such a group identifier to determine a specific one of the groups which has been executed.

34. (canceled)

35. (previously presented) The system of claim 42, wherein said processor groups a sequence of the groups into a block, and wherein said display indicates that said block has been executed only if all of the groups of the block are executed.

36. (original) The system of claim 35, wherein said processor groups said sequence of groups according to a language structure present in said plurality of programs.

37. (original) The system of claim 36, wherein said blocks are defined hierarchically according to the inclusive relationship of language structures in said plurality of programs.

38. (previously presented) The system of claim 42, wherein said processor receives instructions from said input interface to display the statements associated with said unexecuted blocks, said processor causing the statements to be displayed on said display unit such that said tester can determine arguments which would cause an unexecuted block to be execute.

39. (previously presented) The system of claim 38, wherein such an argument comprises an instance of another object.

40. (original) The system of claim 39, wherein said processor instantiates said instance of another object in response to receiving an instruction to instantiate said instance of said

another object, said processor further associating a name associated with said instance of another object, wherein said name is received from said input interface and said tester can enter said name to provide said instance as an argument value.

41. (original) The system of claim 40, wherein said processor receives a string as an argument and determines that said string indicates that said instance is said argument value if said name matches said string.

42. (previously presented) A system enabling a tester to test a program having statements, said computer system comprising:

a random access memory (RAM);

a display unit containing a display screen;

an input interface;

a processor dividing said program into a plurality of groups such that every statement in the program belongs to at least one of the groups, wherein each of said groups contains a respective sequence of ones of the statements such that all the statements of such a group are executed if at least one statement of said group is executed, and wherein such a group is deemed to be executed if at least one of the statements of the group is executed when the program is executed,

said processor executing said program in response to instructions received from said input interface;

said processor determining the ones of the groups that are executed when said program is executed;

said processor causing a display to be generated on said display unit, said display indicating unexecuted ones of the groups based on the ones of the groups that were determined to have been executed;

said processor enabling said tester to execute said unexecuted groups such that said tester can ensure that all statements in said program are executed at least once, wherein said processor includes an extra statement in each of said groups, wherein execution of such an

APPENDIX "AA" CLAIMS

extra statement enables said processor to identify an executed one of the groups corresponding to said extra; and

wherein said computer system further comprises a secondary storage, wherein said processor stores said program including said extra statement on said secondary storage, wherein said program is contained in a plurality of programs which in turn are contained in a class of an object oriented environment, wherein said processor receives a plurality of program lines representing a macro, said processor storing said macro in a database, and said processor executing said macro in response to receiving an instruction to execute said macro.

43. (original) The system of claim 42, wherein said macro is designed to examine the data structures within an instance of an object or to set the values for the variables in the object.

44. (previously presented) The system of claim 42, wherein said processor loads said class into said RAM in response to receiving an instruction to load said class, said processor further instantiating an instance of said class in response to receiving another instruction, said processor executing said program on said instance in response to receiving one more instruction.

45. (previously presented) The system of claim 42, wherein said input interface is connected to at least one of a mouse and a key-board.

APPENDIX "BB" EVIDENCE

NONE.

APPENDIX "CC" RELATED PROCEEDINGS

NONE.

APPENDIX "DD" ATAC REFERENCE

Attached is a copy of the first page of the ATAC reference that was provided to Appellant in the Office action of December 7, 2004. This is enclosed so that it may be seen that the document does not have a publication or copyright date and that it has a November 5, 2004, retrieval date.

[Top] [Prev] [Next] [Index] [TOC]

Chapter 3

ATAC: Overview

This chapter provides an overview of ATAC and is recommended reading for first-time users or those who want a summary of ATAC.

3.1 What is ATAC?

ATAC is a *coverage analysis* tool that aids in testing programs written in the C or C++ programming language. ATAC measures how thoroughly a program has been exercised by a set of tests, identifies code within the program that is not well tested, and determines the overlap among individual test cases. ATAC is used by software developers and testers to measure the *adequacy* of a test set and identify areas in a program that require further testing. These measures may be used for project tracking to indicate progress during testing, and as acceptance criteria to subsequent stages of development and testing. Regression testers also may use ATAC to identify a particular subset of a test set that achieves high coverage at limited cost.

3.2 What is Coverage Testing?

Coverage testing suggests a number of criteria that should be satisfied when testing a program. Examples of such criteria are:

- All statements should be executed;
- All decisions should be evaluated both to true and to false.

The goal of coverage testing is to develop a set of tests that satisfy the criteria. Notice that each of these example criteria are dependent upon a program's source code. Testing methods that use information about a program's internal structure are said to perform *white-box* testing. Methods that only consider a program's inputs and outputs, making no use of its source code, are said to perform *black-box* testing. ATAC supports white-box testing, so the *coverage criteria* discussed here will be tied to the source code of the program under test.

Each specific coverage criterion identifies a number of program constructs to be exercised *(covered)* by a set of tests. The constructs to be covered are called *testable attributes*. For instance, in covering all decisions as suggested above, there is one testable attribute for each true branch and one for each false branch in the program. A tester covers them by developing a test set that executes each of these branches. A test set is considered *adequate* with respect to a given coverage criterion if all testable attributes identified by the criterion are exercised, at least once, by some test within the set.

It is harder to develop an adequate test set for some coverage criteria than for others. Weaker criteria usually require fewer test cases than stronger criteria to obtain completely adequate coverage. However, a test set adequate for a weaker criterion is less likely to